mathematics

MDPI

Article

Optimal Tilt-Wing eVTOL Takeoff Trajectory Prediction Using Regression Generative Adversarial Networks

Shuan-Tai Yeh [1] and Xiaosong Du [2,*]

[1] Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI 48109, USA; shuantai@umich.edu
[2] Department of Mechanical and Aerospace Engineering, Missouri University of Science and Technology, Rolla, MO 65409, USA
* Correspondence: xiaosongdu@mst.edu

Abstract: Electric vertical takeoff and landing (eVTOL) aircraft have attracted tremendous attention nowadays due to their flexible maneuverability, precise control, cost efficiency, and low noise. The optimal takeoff trajectory design is a key component of cost-effective and passenger-friendly eVTOL systems. However, conventional design optimization is typically computationally prohibitive due to the adoption of high-fidelity simulation models in an iterative manner. Machine learning (ML) allows rapid decision making; however, new ML surrogate modeling architectures and strategies are still desired to address large-scale problems. Therefore, we showcase a novel regression generative adversarial network (regGAN) surrogate for fast interactive optimal takeoff trajectory predictions of eVTOL aircraft. The regGAN leverages generative adversarial network architectures for regression tasks with a combined loss function of a mean squared error (MSE) loss and an adversarial binary cross-entropy (BC) loss. Moreover, we introduce a surrogate-based inverse mapping concept into eVTOL optimal trajectory designs for the first time. In particular, an inverse-mapping surrogate takes design requirements (including design constraints and flight condition parameters) as input and directly predicts optimal trajectory designs, with no need to run design optimizations once trained. We demonstrated the regGAN on optimal takeoff trajectory designs for the Airbus A^3 Vahana. The results revealed that regGAN outperformed reference surrogate strategies, including multi-output Gaussian processes and conditional generative adversarial network surrogates, by matching simulation-based ground truth with 99.6% relative testing accuracy using 1000 training samples. A parametric study showed that a regGAN surrogate with an MSE weight of one and a BC weight of 0.01 consistently achieved over 99.5% accuracy (denoting negligible predictive errors) using 400 training samples, while other regGAN models require at least 800 samples.

Keywords: eVTOL; optimal takeoff trajectory design; machine learning; surrogate modeling; generative adversarial networks; regGAN; inverse mapping

MSC: 93C85

Citation: Yeh, S.-T.; Du, X. Optimal Tilt-Wing eVTOL Takeoff Trajectory Prediction Using Regression Generative Adversarial Networks. *Mathematics* **2024**, *12*, 26. https://doi.org/10.3390/math12010026

Academic Editor: Daniel-Ioan Curiac

Received: 29 November 2023
Revised: 15 December 2023
Accepted: 19 December 2023
Published: 21 December 2023

1. Introduction

The attention on electric vertical takeoff and landing (eVTOL) aircraft has grown significantly because eVTOL is suitable for missions that require flexible maneuverability and precise control, such as urban air mobility (UAM) [1,2]. UAM represents a safe and efficient air transportation system where everything from small package delivery drones using unmanned aerial vehicles (UAVs) to passenger-carrying air taxis through eVTOL aircraft can operate above populated areas [3,4]. UAVs have broad applications, including the sixth-generation communication using UAVs and an aerial base station for supporting the Internet of Things deployment in remote and disaster areas [5]. Hua et al. [6] proposed 3D non-stationary modeling for UAV-to-ground communications and managed to demonstrate its validity and practicality against measured results. eVTOL aircraft's

unique characteristics of precise delivery, lower cost, and reduced noise have motivated significant developments [7,8]. Conceptions under development can be categorized into a few major aircraft types, including lift+cruise (such as Aurora Flight Sciences eVTOL [9]) and tilt-wing (such as Airbus A^3 Vahana [10]).

Among the aforementioned eVTOL types, the tilt-wing configuration enables the aircraft to combine the flexibility of a helicopter for vertical takeoff and landing with the efficiency of airplanes during cruising. Thus, the transition optimization on tilt-wing eVTOL aircraft attracts special interest since transition has major effects on the success of flight tasks [11]. Pradeep and Wei [12,13] put an emphasis on the formulation of multiphase optimal control problems with energy consumption index for tilt-wing and multirotor eVTOL vehicles. The proposed multiphase optimal control problem formulation and the numerical solution allowed an eVTOL air taxi to fulfill the specified required arrival time while attaining the most energy-efficient trajectory for arrival. This capability played a vital role in enabling safe and efficient future operations of eVTOL aircraft, facilitating passenger transportation and cargo delivery. Chauhan and Martins [14] constructed an Airbus A^3 Vahana [10] model and optimized its takeoff-to-cruise trajectory with the objective of minimizing energy consumption. They concluded that the optimal takeoff trajectory involved stalling the wings or flying near the stall angle of attack. Moreover, in the absence of acceleration constraints, the optimized trajectories involved a rapid transition to forward flight, followed by climbing at a relatively constant speed, and then accelerating to the desired cruising speed. When considering passenger comfort and incorporating acceleration constraints, the transition, climb, and acceleration phases exhibited more gradual and less distinct behavior, as expected. However, completed work on transition optimization as well as conventional engineering design optimizations rely on iterative simulation model evaluations, which prohibit real-time decision making.

Thus, surrogate models have emerged as an effective alternative for fast interactive decision making in lieu of time-consuming simulation models [15,16]. Surrogate-based design optimization is a methodology that leverages surrogate models to approximate the behavior of design candidates. To elaborate, surrogate models are mathematical models that are trained using a limited number of data points obtained from computationally expensive simulations or physical experiments. These surrogate models are then used in optimization algorithms to efficiently make predictions, explore design space, and find optimal solutions. Surrogate-based design optimization is advantageous over conventional simulation-based optimization for computational efficiency [17,18], the effective exploration of high-dimensional design spaces [19,20], multi-objective optimization capabilities [21,22], sensitivity analysis [23], and uncertainty analysis [24–26].

The Gaussian process (GP) [27] is one of the most commonly used surrogate models due to its capability and flexibility for modeling and predicting unknown functions based on observed data [28]. The concept of a multi-output GP (MOGP) has emerged to further handle problems with multiple outputs [29,30]. By extending the concept of a GP, an MOGP is a surrogate modeling technique that allows for the joint modeling of multiple correlated outputs and the simultaneous capture of their dependencies. In particular, by jointly modeling the outputs by estimating the covariance matrix (also known as the correlation function), an MOGP leverages the shared information among the outputs, which reduces the overall modeling complexity. For more mathematical and surrogate modeling details, please refer to Section 2.3. Meanwhile, deep learning surrogates are going through revolutionary developments and pushing forward cutting-edge research in the design optimization community. Thelen et al. [31] computed design variable derivatives by analytically linking objective definition, mesh, and geometry, and they demonstrated the derivatives through the multi-fidelity Broyden–Fletcher–Goldfarb–Shannon algorithm for high-dimensional aerodynamic and aeroelastic design optimizations. Tao and Sun [32] developed a multi-fidelity surrogate-based optimization framework based on deep belief networks. The results for airfoil and wing designs under uncertainty indicated that the multi-fidelity surrogate model performed well by significantly improving optimization

efficiency. Renganathan et al. [33] performed aerodynamic design optimization by fusing deep neural networks (DNN) and GP as one surrogate to incorporate the predictive power of DNN and the confidence interval of GP. Thus, the DNN-GP surrogate enabled relatively high-dimensional Bayesian optimization on aerodynamic design and outperformed adjoint-based optimization in terms of efficiency.

Even with these developments, surrogate modeling still cannot completely address the "curse of dimensionality", which necessitates dimensionality reduction techniques for further improvements. O'Leary-Roseberry et al. [34] constructed adaptive residual networks on reduced-dimensional space exploited by principal component analysis to directly predict optimal designs based on design requirements. They successfully demonstrated outstanding performance over full-space feed-forward networks on aerodynamic wing design cases. Meanwhile, a prosperous family of machine learning models, generative adversarial networks (GAN), was developed as generative models [35] with the feature of automatic design space reduction. The competition between a generator and a discriminator allows them to generate new data that follow the same patterns as the training data. Therefore, when fed with realistic airfoil or wing design shapes, a GAN model enables implicit design space dimensionality reduction by filtering out unrealistic shapes and generating only realistic designs [36]. Du et al. [37] developed the B-spline-based generative adversarial networks (BSplineGAN) for intelligent airfoil parameterization with the UIUC airfoil database as training data. BSplineGAN automatically reduced the design space while maintaining sufficient shape variability, which was verified by fitting optimizations to arbitrary UIUC airfoils. They constructed DNN surrogates on the reduced space exploited by BSplineGAN and verified predictive performance on aerodynamic design cases in a fast, interactive manner.

Derived from the original GAN, GAN variants also handle regression tasks. Du and Martins [38] introduced a novel multi-fidelity surrogate modeling architecture, super-resolution GAN (SRGAN), for predicting airfoil pressure distributions based on low-fidelity counterparts. Specifically, the SRGAN generator generated super-resolution pressure distributions, while the discriminator aimed to distinguish between pressure distributions of the generated super-resolution shapes and the high-resolution data set. Thus, training the generator minimized the difference between the super-resolution shapes and corresponding high-fidelity (i.e., high-resolution) data and maximized the similarity of super-resolution with the high-fidelity data. The results showed that the SRGAN outperformed low-fidelity simulations and direct DNN by accurately capturing the locations and magnitudes of strong high-fidelity shocks. A conditional generative adversarial networks (cGAN) incorporates additional conditioning information into the generative process [39]. The goal of a cGAN is to generate samples that not only resemble the real data but also adhere to specified conditions. The conditioning information serves as a guide for the generator to generate samples that align with the desired conditions. Aggarwal et al. [40] conducted experiments demonstrating the effectiveness of a cGAN model for regression problems. The cGAN was successfully demonstrated on a real-world ailerons data set for an F-16 airplane. The cGAN managed to predict the control input on the aircraft's ailerons, which was described by 40 continuous inputs. Ye et al. [41] proposed a novel GAN-based regression model (regGAN), adopting a combined loss function on a mean squared error (MSE) and a binary cross-entropy (BC) loss, which showed outstanding predictive performance on frying oil deterioration when provided with time and temperature as input parameters.

Prior research has demonstrated the potential of machine learning surrogates, including GAN variants. The SRGAN and cGAN exhibited predictive potential, while the regGAN realized predictions from physical input space to output space. We demonstrate the regGAN performance in predicting optimal eVTOL takeoff trajectories directly based on design requirements and compare results with the MOGP and cGAN surrogates. We summarize the contribution of this paper as follows. First, we develop and introduce the regGAN surrogate into the takeoff trajectory design area for the first time. Second, we realize the surrogate-based optimal takeoff trajectory design for eVTOL aircraft to fill the

lack of research study in this field. Third, we introduce the inverse mapping concept (from design requirements, including design constraints and flight condition parameters, directly into optimal designs) to eVTOL trajectory design for the first time.

We organize the rest of the paper as follows. Section 2 introduces the optimization framework and simulation models used in this work, followed by MOGP, cGAN, and regGAN setups for surrogate modeling. We demonstrate the regGAN as well as other surrogates on eVTOL optimal takeoff trajectory predictions in Section 3. We end this paper with conclusions in Section 4.

2. Methodology

In this section, we introduce the open-source optimization toolbox `Dymos` (https://github.com/OpenMDAO/RevHack2020/tree/master/problems/evtol_trajectory/evtol_dymos, accessed on 1 August 2022) [42] within `OpenMDAO` (https://github.com/OpenMDAO/OpenMDAO, accessed on 1 August 2022) [43] and simulation models. Additionally, we detail an MOGP toolkit (`MOGPTK` (https://github.com/GAMES-UChile/mogptk, accessed on 1 August 2022)) [44] followed by DNN and GAN series models (namely, GAN, cGAN, and regGAN) within `TensorFlow` [45] for surrogate modeling.

2.1. *Dymos Framework*

`OpenMDAO` [43], an open-source framework for multidisciplinary design, analysis, and optimization (MDAO), empowers engineers and scientists to efficiently analyze and optimize complex engineering systems. It is primarily designed for gradient-based optimization, which is based on efficient coupled derivative computation and problem sparsity exploitation [43]. Its component-based architecture enables the integration of models from diverse disciplines, facilitating seamless collaboration and reusability. `OpenMDAO` excels in interdisciplinary coupling, managing data flow, solving systems of equations, and offering efficient automatic differentiation for optimization and sensitivity analysis. Additionally, `OpenMDAO` supports parallel computing, leveraging multiple cores and distributed computing clusters for efficient analyses.

`Dymos` [42] is an advanced software toolkit built upon the `OpenMDAO` framework by the National Aeronautics and Space Administration for optimizing complex aerospace systems. `Dymos` expands `OpenMDAO` capabilities by offering specialized tools and algorithms for dynamic optimization. With a primary focus on dynamic systems, `Dymos` transcends solving problems that involve the evolution of systems over time. This is valuable for trajectory optimizations, where various constraints and objectives need to be considered. By employing numerical methods, such as direct transcription and indirect shooting methods, `Dymos` can discretize and optimize the trajectory based on specified nodes or iteratively refine initial guesses, respectively. The software also provides an intuitive interface and visualization tools for easy problem formulation and manipulation while supporting integration with external software and models. `Dymos` serves as a powerful and versatile toolkit for dynamic optimization and multidisciplinary analysis, enabling enhanced design and optimization of complex aerospace systems.

In this work, we utilize one of the implicit collocation techniques demonstrated in `Dymos`: higher-order Gauss–Lobatto collocation (LGL). Implicit collocation techniques are especially suitable for gradient-based optimization including analytic derivatives [42]. The equation of motion propagates explicitly through time-marching and requires a fixed number of analysis points in time with enough density to capture the system dynamics accurately. Implicit collocation schemes are able to attain this with fewer analysis points in a grid fixed throughout an optimization. In `Dymos`, implicit collocation methods further disintegrate each phase into one or more segments. For each segment, to ensure the value in time can be modeled as a polynomial, each state or control variable is assumed to be C^1 continuous. Additionally, within each segment, state and control values are allocated at specific points or nodes, indicated by the state discretization nodes and control discretization nodes, respectively. Inside each phase, there are several variables described

in the following, such as design variables (initial time t_0, duration t_p), state variables at the state discretization nodes ($\bar{x}_d$), dynamic control variables at the control discretization nodes ($\bar{u}_d$), and the design parameters ($\bar{d}$). Herman and Conway proposed the high-order LGL method, which is the primary method used in OTIS [46] and is a general extension of the Hermite-Simpson collocation to higher orders [47]. In a phase, the segments are discretized at the LGL nodes so that, for each segment, the total number of nodes is an odd value. The state discretization nodes are the nodes with an even index, where the index starts at zero for the first node. Dynamic controls are provided at designated control discretization nodes. In LGL collocation, all nodes within the phase are involved in the control discretization subset. The evaluation of an LGL phase in `Dymos` progresses with the following steps: Initially, the values of the design variables (t_0, t_p, $\bar{x}_d$, $\bar{u}_d$, and $\bar{d}$) are given by the optimizer and user. Secondly, the `OpenMDAO` system offering the state dynamics is assessed at the state discretization nodes, contributing the state time derivatives at the state discretization nodes:

$$\dot{\bar{x}}_d = f_{\text{ode}}(t_d, \bar{x}_d, \bar{u}_d, \bar{d}), \tag{1}$$

where f_{ode} is the governing ordinary differential equation (ODE) for simulation models.

Then, using Hermite interpolation [48], the states and state rates are interpolated to the collocation nodes:

$$\bar{x}_c = [A_i]x_d + \frac{t_{\text{seg}}}{2}[B_i]\dot{\bar{x}}_d, \tag{2}$$

$$\bar{x}'_c = \frac{2}{\text{seg}}[A_d]x_d + [B_d]\dot{\bar{x}}_d, \tag{3}$$

where t_{seg} is the duration of the segment where each nodes originates. At the collocation nodes, the state system provided by the system at this point is evaluated and is given by

$$\dot{\bar{x}}_c = f_{\text{ode}}(t_c, \hat{x}_c, \hat{u}_c, d). \tag{4}$$

Consequently, there are two independent state rate values existing: one is the actual value derived from the ODE, and the other is the approximate value calculated from the slope of the interpolating polynomials. For each state in the phase, the "collocation defects, Δ", normalized by segment time space, consists of the difference between two state rates. Moreover, these values are constrained at zero for a nonlinear programming problem and can be expressed as

$$\bar{\Delta} = \bar{x}'_c - \frac{\bar{t}_{\text{seg}}}{2}\dot{\bar{x}}_{\text{col}}. \tag{5}$$

2.2. Simulation Models

We consider the verified simulation models covering aerodynamics, propulsion, propeller–wing interaction, and dynamics developed by the `Dymos` team [42].

2.2.1. Aerodynamics

The aerodynamics model assumes that the forward and rear wings are equivalent and have the same reference area for simplicity, and there is no flow interaction between the two wings. The rotations of the two wings are assumed to be the same, so that the angles of attack are equivalent as well as the lift and drag generated by the two wings. During the transition from vertical to horizontal flight, separated-flow conditions are considered. A model developed by Tangler and Ostowari [49] for wing aerodynamics is implemented to predict the lift and drag beyond the linear-lift region. The poststall lift coefficient is

$$C_L = A_1 \sin 2\alpha + A_2 \frac{\cos^2 \alpha}{\sin \alpha}, \tag{6}$$

where

$$A_1 = \frac{C_1}{2}, \tag{7}$$

$$A_2 = (C_{L_s} - C_1 \sin\alpha_s \cos\alpha_s) \frac{\sin\alpha_s}{\cos^2\alpha_s}, \tag{8}$$

and

$$C_1 = 1.1 + 0.018AR, \tag{9}$$

where α is the wing angle of attack (in radians), α_s is the angle of attack at stall (in radians), C_{L_s} is the lift coefficient at stall, and AR is the wing aspect ratio.

The drag coefficient at a wing angle of attack between 27.5 and 90 degrees is given by

$$C_D = B_1 \sin\alpha + B_2 \cos\alpha, \tag{10}$$

where

$$B_1 = C_{D_{\max}}, \tag{11}$$

$$B_2 = \frac{C_{D_s} - C_{D_{\max}} \sin\alpha_s}{\cos\alpha_s}, \tag{12}$$

and

$$C_{D_{\max}} = \frac{1.0 + 0.065AR}{0.9 + t/c}, \tag{13}$$

where C_{D_s} is the drag coefficient at stall, and t/c is the airfoil thickness-to-chord ratio. For the poststall drag coefficient at a wing angle of attack below 27.5 degrees, the equation is

$$C_D = 0.008 + 1.107\alpha^2 + 1.792\alpha^4, \tag{14}$$

where α is in radians.

The well-known finite wing corrections from lifting-line theory for unswept wings in incompressible flow are utilized to modify the lift and drag of the airfoil prior to stalling,

$$\alpha_{\text{wing}} = \frac{\alpha_{\text{airfoil}}}{1 + (\alpha_{\text{airfoil}}/(\pi \cdot AR \cdot e))}, \tag{15}$$

where α_{wing} is the finite-wing lift–curve slope, α_{airfoil} is the airfoil lift–curve slope, and e is the span efficiency factor. The prestall lift curve is assumed to be linear, and the wing stall angle of attack is 15 deg.

In order to obtain the total drag of the wing before stall, induced drag has been added using the well-known formula based on the lifting-line theory for prestall parasite drag coefficients,

$$C_{D_i} = \frac{C_L^2}{\pi \cdot AR \cdot e}, \tag{16}$$

where C_{D_i} is the induced drag coefficient, C_L is the wing's lift coefficient, and $AR = 8$ for each wing of the configuration.

Additional drag on an assumed drag area for the fixed landing gear and the fuselage is assumed to be independent of the freestream angle of attack. The induced drag for the configuration can be expressed as

$$D_{\text{induced}} = 2\left(1.4\frac{L_{\text{wing}}^2}{\pi q b^2 \cdot 0.95}\right) = 2\left(\frac{L_{\text{wing}}^2}{\pi q b^2 \cdot 0.68}\right), \tag{17}$$

where L_{wing} is lift per wing, q is freestream dynamic pressure, b is wing span.

2.2.2. Propulsion

Momentum theory [50] has been utilized to compute thrust from the propellers as a function of power.

$$P_{\text{disk}} = TV_{\infty\perp} + \kappa T\left(-\frac{V_{\infty\perp}}{2} + \sqrt{\frac{V_{\infty\perp}^2}{4} + \frac{T}{2\rho A_{\text{disk}}}}\right), \tag{18}$$

where P_{disk} is the power supplied to the propeller disk excluding profile power, T is the thrust, $V_{\infty\perp}$ is the freestream velocity component normal to the propeller disk, ρ is the air density, A_{disk} is the disk area of the propeller, and κ is the correction factor utilized to incorporate induced power losses associated with non-uniform flow, tip effects, and other simplifications made in momentum theory ($\kappa = 1$ for ideal power). Power is used as a design variable in the optimization problems, and the Newton–Raphson method [51] has been chosen to solve the nonlinear equation for thrust with power as an input.

Applying blade-element theory to a rotor operating in nonaxial forward flight [52,53] estimates the profile power coefficient C_{P_p} and profile power P_p

$$C_{P_p} = \frac{\sigma C_{d_{0p}}}{8}(1 + 4.6\mu^2), \tag{19}$$

and

$$C_{P_p} = \frac{P_p}{\rho A_{\text{disk}} R^3 \Omega^3}, \tag{20}$$

where R is the radius of the propeller, Ω is the angular speed, σ is the solidity, $C_{d_{0p}}$ is a representative constant profile drag coefficient, and μ is defined as

$$\mu = \frac{V_{\infty\parallel}}{\Omega R}, \tag{21}$$

where $V_{\infty\parallel}$ is the freestream velocity component parallel to the disk. For our cases, we assume that $\Omega = 181$ rad/s for $R = 0.75$ m, $\sigma = 0.13$, and $C_{d_{0p}} = 0.012$.

With electrical power as an input, we use a factor k_{elec} to account for electrical and mechanical losses related to batteries, electrical systems, and motors. P_{disk} can be expressed as

$$P_{\text{disk}} = k_{\text{elec}} P_{\text{electrical}} - P_p, \tag{22}$$

where $P_{\text{electrical}}$ is the power from the batteries. We set the limit on the maximum electrical power available to 311 kW. The loss factor k_{elec} ranges from 0.7 to 0.9 and is also one of our design requirements considered as input parameters for surrogate modeling.

Furthermore, when the freestream flow is not completely normal to the propeller disks, the normal force N is calculated as

$$N = \frac{4.25\sigma_e \sin(\beta + 8^\circ) f q_\perp A_{\text{disk}}}{1 + 2\sigma_e} \tan \alpha_{in}, \tag{23}$$

where $q_\perp$ is the dynamic pressure based on the freestream velocity component normal to the propeller disk, β is the blade pitch angle at $0.75R$ and is assumed to change linearly from 10 deg at a flight speed of 0 m/s to 35 deg at a cruise speed of 67 m/s, A_{disk} is the propeller area, and α_{in} is the incidence angle. The remaining terms are calculated in the following manner. The effective solidity σ_e is

$$\sigma_e = \frac{2Bc_{\text{b}}}{3\pi R}, \tag{24}$$

where B is the number of blades per propeller, c_{b} is the average chord length of the blades, and R is the propeller radius. The thruster factor f is

$$f = 1 + \frac{\sqrt{1+T_c}-1}{2} + \frac{T_c}{4(2+T_c)}, \tag{25}$$

where T_c is a thrust coefficient, defined as

$$T_c = \frac{T}{q_\perp A_{\text{disk}}}, \tag{26}$$

and T is the thrust.

2.2.3. Propeller–Wing Interaction

The momentum theory [50] is implemented for modeling the interaction between a wing and the flow induced by propellers. The induced speed at the disk v_i is given by

$$v_i = \left(-\frac{V_{\infty\perp}}{2} + \sqrt{\frac{V_{\infty\perp}^2}{4} + \frac{T}{2\rho A_{\text{disk}}}} \right). \tag{27}$$

To explain the propeller–wing interaction in this work, the chordwise component of the freestream velocity for the entire wing by a range of factors k_{in} is varied between 0.3 and 1, multiplied by the induced speed at the disk v_i. The effective k_{in} for the wing is anticipated to be close to 1 but slightly less than it, especially when the aircraft is at low speed. Moreover, cases will become infeasible if k_{in} is too low; therefore, we set the range $k_{\text{in}} \in [0.3, 1.0]$ in this paper to make sure all cases are feasible under the design requirements.

2.2.4. Dynamics

A two degrees of freedom (DOF) representation and the forward Euler method [54] are used for simulating the trajectory of the aircraft. The horizontal and vertical components of the aircraft velocity are solved as functions of time, considering the control variables, which include the wing-tilt angle and electrical power. The horizontal component of velocity at each time step is calculated as follows:

$$v_{x_{i+1}} = v_{x_i} + \frac{T\sin\theta - D_{\text{fuse}}\sin(\theta+\alpha_\infty) - D_{\text{wings}}\sin(\theta+\alpha_{\text{EFS}}) - L_{\text{wings}}\cos(\theta+\alpha_{\text{EFS}}) - N\cos\theta}{m}\Delta t, \tag{28}$$

where i is index of a time step, Δt is a time step length, θ is a wing angle relative to the vertical, α_∞ is a freestream angle of attack, α_{EFS} is the effective freestream angle of attack experienced by the wings due to the propeller influence, m is the mass of the aircraft, T is the total thrust, D_{fuse} is the drag of the fuselage, D_{wings} is the total drag of the two wings, L_{wings} is the total lift of the two wings, and N is the total normal force generated by the propellers. Figure 1 displays the forces and the angles on the aircraft. Likewise, the vertical component of velocity at each time step is calculated as follows:

$$v_{y_{i+1}} = v_{y_i} + \frac{T\cos\theta - D_{\text{fuse}}\cos(\theta+\alpha_\infty) - D_{\text{wings}}\cos(\theta+\alpha_{\text{EFS}}) + L_{\text{wings}}\sin(\theta+\alpha_{\text{EFS}}) + N\sin\theta - mg}{m}\Delta t, \tag{29}$$

where g is the gravitational acceleration. To determine the time step length, Δt, the total takeoff time, which is a design variable in optimization problem formulation, is divided by 500. A convergence study concluded the optimized takeoff flight time would be typically below 50 s, resulting in adequately small time steps of less than 0.1 s [14].

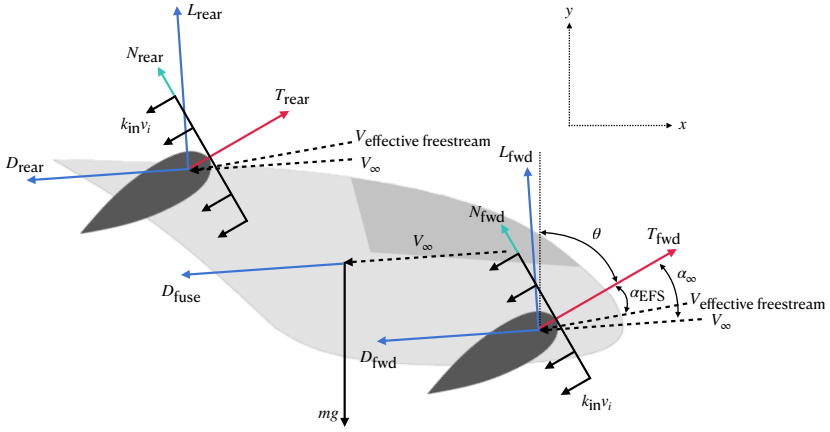

Figure 1. Definitions of angle and forces on the aircraft.

2.3. Multi-Output Gaussian Processes

A GP is a Bayesian nonparametric model that is designed by parameterizing a covariance kernel, meaning that constructing expressive kernels allows for an effective representation of complex systems [44]. The GP assumes the function $f(\mathbf{x})$ is distributed as [55]

$$f(\mathbf{x}) \sim \mathcal{GP}\big(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')\big), \tag{30}$$

where a Gaussian process is a distribution over functions and is characterized by a mean function $m(\mathbf{x})$ and a covariance function $k(\mathbf{x}, \mathbf{x}')$ between input parameters $\mathbf{x}$ and $\mathbf{x}'$. The mean function represents the expected function value at input $\mathbf{x}$

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]. \tag{31}$$

In other words, the mean function represents the average function value at input $\mathbf{x}$ across all functions in the distribution. To simplify posterior computations and rely only on the covariance function for inference, the prior mean function is frequently assumed to be a constant. In this work, we apply the constant mean function ($m(\mathbf{x}) = b_1$) and the linear mean function ($m(\mathbf{x}) = a\mathbf{x} + b_2$), where a, b_1, and b_2 are arbitrary constant coefficients. The covariance function $k(\mathbf{x}, \mathbf{x}')$ captures the relationship between the function values at different input points $\mathbf{x}$ and $\mathbf{x}'$:

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}\big[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))\big]. \tag{32}$$

The function k is commonly called the kernel of the Gaussian process. The kernel defines the shape and characteristics of the GP model. We implement the Matérn kernel and the square exponential kernel function in our study, and they are described as follows.

1. Matérn kernel
   The Matérn covariance between two points with the distance $\tau = |\mathbf{x} - \mathbf{x}'|$ is

$$k_\nu(\tau) = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu}\frac{\tau}{\rho}\right) K_\nu \left(\sqrt{2\nu}\frac{\tau}{\rho}\right), \tag{33}$$

   where Γ is the gamma function, K_ν is the modified Bessel function of the second kind, and ρ and ν are non-negative covariance parameters. The Matérn kernel is stationary since the covariance only depends on distances between points.
2. Squared exponential kernel

$$k_{SE} = s^2 \exp\left(-\frac{1}{2}\left(\frac{|\mathbf{x} - \mathbf{x}'|}{\lambda}\right)\right), \tag{34}$$

where s is the scale factor. In the case where we use a single length scale parameter, the squared exponential (SE) kernel is an example of a radial basis function.

The GP has been extended to multiple series (or channels), which are referred to as MOGP. The key difference lies in their implementation and capabilities. The GP is designed for single-output regression, predicting a single target variable given inputs, while an MOGP extends the GP to handle multiple outputs simultaneously. An MOGP is useful when dealing with correlated outputs, enabling the modeling of dependencies between different target variables. The complexity of a GP relates to the number of data points, while an MOGP introduces additional complexity due to modeling correlations between multiple outputs, but the MOGP remains efficient and more capable compared with training individual GPs for each output. Thus, a key feature of the MOGP is to harness applicable information across outputs to provide more accurate predictions than separately modeling correlated outputs [28].

In this study, the multi-output spectral mixture [56] kernel is used as the multivariate kernel for the MOGP model, and can be expressed as

$$k_{ij}(\tau) = \sum_{q=1}^{Q} \alpha_{ij}^{(q)} \exp\left(-\frac{1}{2}\left(\tau + \theta_{ij}^{(q)}\right)^{\top} \Sigma_{ij}^{(q)} \left(\tau + \theta_{ij}^{(q)}\right)\right) \cos\left(\left(\tau + \theta_{ij}^{(q)}\right)^{\top} \mu_{ij}^{(q)} + \phi_{ij}^{(q)}\right), \quad (35)$$

where $\alpha_{ij}^{(q)} = w_{ij}^{(q)} (2\pi)^{\frac{n}{2}} \left|\Sigma_{ij}^{(q)}\right|^{1/2}$ and the superindex $(\cdot)^{(q)}$ denotes the parameter of the q^{th} component of the spectral mixture. In addition, the cross-spectral density between channels i and j is modeled as a complex-valued SE function with the following parameters:

- Covariance: $\Sigma_{ij} = 2\Sigma_i \left(\Sigma_i + \Sigma_j\right)^{-1} \Sigma_j$;
- Mean: $\mu_{ij} = \left(\Sigma_i + \Sigma_j\right)^{-1} \left(\Sigma_i \mu_j + \Sigma_j \mu_i\right)$;
- Magnitude: $w_{ij} = w_i w_j \exp\left(-\frac{1}{4} \left(\mu_i - \mu_j\right)^{\top} \left(\Sigma_i + \Sigma_j\right)^{-1} \left(\mu_i - \mu_j\right)\right)$;
- Delay: $\theta_{ij} = \theta_i - \theta_j$;
- Phase: $\phi_{ij} = \phi_i - \phi_j$;

where the so-constructed magnitudes w_{ij} ensure positive definiteness.

We use the `MOGPTK` [44] to establish MOGP models. `MOGPTK` is an open-source Python package that provides a natural way to train and use the models. `MOGPTK` is built upon GPflow [57], an extensive GP framework with a wide variety of implemented kernels, likelihoods, and training strategies. The main components of `MOGPTK` include MOGP modeling, data handling, parameter initialization, and parameter interpretation [44].

2.4. Deep Neural Networks

The DNN makes the core of deep learning. In this section, we introduce the DNN model basic setup and the Adam optimizer used for model training.

2.4.1. DNN Model Setup

A DNN [58] consists of multiple layers of interconnected nodes, known as neurons, to process input data and make predictions. A DNN starts with an input layer (reading input data), followed by one or more hidden layers, each of which consists of multiple neurons. The final hidden layer connects to the output layer, which produces the predictions. Moreover, neurons in the same layers share a weight matrix ($\mathbf{W}$) and a bias vector ($\mathbf{b}$) to be tuned. The equation within each neuron can be mathematically expressed as

$$\mathbf{o} = \sigma(\mathbf{W} \cdot \mathbf{h} + \mathbf{b}), \quad (36)$$

where $\mathbf{h}$ is a data vector from previous layers, $\sigma(\cdot)$ represents an activation function used in the current layer, and $\mathbf{o}$ is an output vector of the current layer. During training, $\mathbf{W}$ and $\mathbf{b}$ are adjusted to optimize the model's performance. We utilize two commonly used activation functions in this work to introduce nonlinearity into the DNN. The rectified

linear unit (`relu`) [59] activation function follows the equation $\sigma_{\texttt{relu}}(\mathbf{h}) = \max(\mathbf{0}, \mathbf{h})$, while the `sigmoid` [60] activation function has an expression of $\sigma_{\texttt{sigmoid}}(\mathbf{h}) = 1/\left(1 + e^{-\mathbf{h}}\right)$.

In this work, we harness two types of loss functions for training DNN surrogates to minimize the difference between predicted responses and corresponding actual values. The MSE [61] loss function is formulated as

$$\mathcal{L}_{\text{MSE}} = \frac{1}{N}\sum_{i=1}^{N}(|\mathbf{v}_i - \hat{\mathbf{v}}_i|_2)^2, \tag{37}$$

where N is the number of training data points in the data set, $|\cdot|_2$ is L_2 norm, $\mathbf{v}_i$ and $\hat{\mathbf{v}}_i$ are actual values and predicted values for the ith training sample. Binary crossentropy (BC) [62] is formulated as

$$\mathcal{L}_{\text{BC}} = -\frac{1}{N}\sum_{i=1}^{N}(\mathbf{v}_i \cdot \log \hat{\mathbf{v}}_i + (\mathbf{1} - \mathbf{v}_i) \cdot \log(\mathbf{1} - \hat{\mathbf{v}}_i)), \tag{38}$$

where $\mathbf{v}_i$ is the ith actual binary label (**0** or **1**), and $\hat{\mathbf{v}}_i$ is the predicted probability (between **0** and **1**). The loss function penalizes the model more severely when it predicts the opposite class with high confidence. In this work, the DNN unknown parameters are tuned by minimizing loss functions through the gradient-based Adam optimizer (see the following section) enabled by backpropagation within `Tensorflow` [45].

2.4.2. Adam Optimizer

The Adam optimizer combines the principles of gradient descent with momentum [63] and root mean square propagation (RMSP) [64] algorithms [65]. The gradient descent algorithm has been enhanced by the moment algorithm [63], with the updates as

$$w_{t+1} = w_t - \alpha m_t, \tag{39}$$

where

$$m_t = \beta m_{t-1} + (1 - \beta)\left[\frac{\partial L}{\partial w_t}\right], \tag{40}$$

where the subscripts $t-1$, t, and $t+1$ are the indices of previous, current, and next optimization steps, respectively. Initially, $m_0 = 0$, w is a weight to be determined, α is learning rate, $\frac{\partial L}{\partial w_t}$ is the derivative of loss function with respect to weight at current optimization step, and β is a constant moving average parameter.

RMSP [64] is an adaptive learning algorithm with a mathematical formulation as

$$w_{t+1} = w_t - \frac{\alpha}{(v_t + \epsilon)^{1/2}}\left[\frac{\partial L}{\partial w_t}\right]^2, \tag{41}$$

where

$$v_t = \beta v_{t-1} + (1 - \beta)\left[\frac{\partial L}{\partial w_t}\right]. \tag{42}$$

A small positive constant ($\epsilon = 10^{-7}$) is utilized. The remaining variables remain the same as in Equation (40).

The Adam optimizer integrates the moment and RMSP algorithms as follows.

$$w_{t+1} = w_t - \hat{m}_t\left(\frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon}\right), \tag{43}$$

where

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t}, \tag{44}$$

where m_t and v_t follow the updating process as described above.

2.5. Generative Adversarial Network Models

A GAN is a novel DNN architecture that is leading in state-of-the-art generative modeling strategies. This section introduces GAN setups along with their variations, namely, cGAN and regGAN.

2.5.1. Generative Adversarial Networks

A GAN is in nature a type of generative model that consists of two neural networks: a generator and a discriminator (Figure 2) [35]. The generator initially produces random outputs based on random input variables, but as it receives feedback from the discriminator, it learns to generate more realistic samples through training. The discriminator, on the other hand, is trained on a data set of existing samples and the generated samples created by the generator. The discriminator is to distinguish between existing and generated data and provide feedback to the generator by assigning probabilities to the existing and generated samples, indicating how likely they are to be real. During the training process, the generator and the discriminator attempt to compete with each other in an adversarial manner. This competition drives the improvement of both models through the model training process. Thus, as the training proceeds, the generator is more capable of producing similar samples as existing data that the discriminator cannot differentiate.

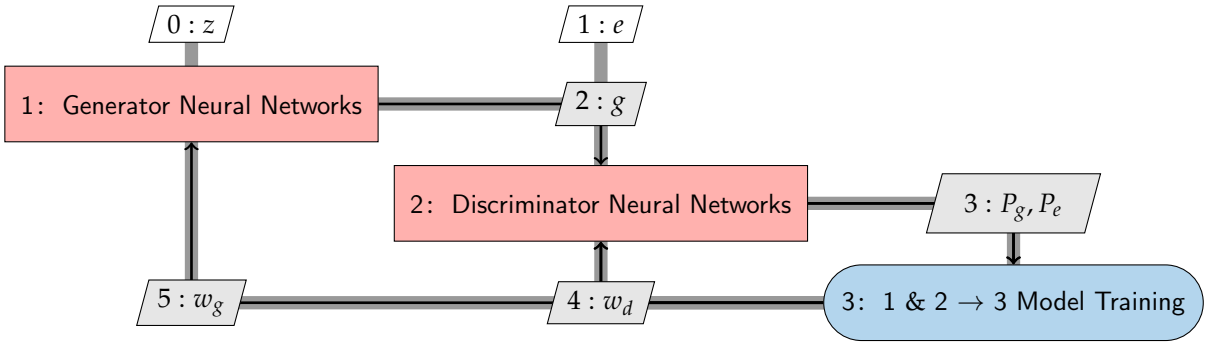

Figure 2. GAN contains the discriminator to compete with the generator. The generator generates shapes (**g**) based on random variables (**z**), typically following user-defined distributions (uniform distribution in this work). The discriminator distinguishes between the existing data (**e**) and the generated data (**g**). During training, the discriminator adjusts its weights ($\mathbf{w}_d$) to make the probability $\mathbf{p}_g$ of **g** approach zero's while increasing the probability $\mathbf{p}_e$ of **e** towards one's. Conversely, during training, the generator adjusts its weights ($\mathbf{w}_g$) to increase the probability of $\mathbf{p}_g$ towards one's. The generator, after training, generates shapes similar to the existing data.

Mathematically, training a GAN model can be expressed as a minimax problem [35] on the loss function as follows.

$$\min_G \max_D \mathcal{L}_{\text{GAN}}(D, G) = \mathbb{E}_{\mathbf{x}\sim P_{\text{data}}}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim P_{\mathbf{z}}}[\log(1 - D(G(\mathbf{z})))], \tag{45}$$

where **x** is sampled from the existing data distribution P_{data}, **z** is sampled from the noise variable distribution $P_{\mathbf{z}}$, and G and D are the generator and discriminator. Thus, a trained GAN model generates reasonable designs with ample shape variability within prior noise variable distributions.

2.5.2. Conditional Generative Adversarial Networks

A cGAN is an extension of the original GAN that incorporates additional conditioning information into the generative process [39]. For the generator and the discriminator in cGANs, both networks receive not only random noises as input but also additional conditioning information **y**. This information offers extra guidance to the generator to produce samples that align with specified conditions. Conditioning information **y** can be in several

forms, such as text descriptions and class labels, depending on the utilization. The conditioning is achieved by providing $\mathbf{y}$ as an additional input layer to both the discriminator and generator. Within the generator, the prior input noise $P_{\mathbf{z}}(\mathbf{z})$ and $\mathbf{y}$ are combined in a joint hidden representation, benefiting from the adversarial training framework's flexibility in composing this hidden representation. In addition, the discriminator takes as inputs $\mathbf{x}$ and $\mathbf{y}$ pairs as well as shapes generated by the generator to complete the adversarial competition with the generator. Similarly, as a GAN loss function (Equation (45)), the training on loss functions could also be expressed as a minimax problem,

$$\min_G \max_D \mathcal{L}_{\text{cGAN}}(D,G) = \mathbb{E}_{\mathbf{x}\sim P_{\text{data}}(\mathbf{x})}[\log D(\mathbf{x}|\mathbf{y})] + \mathbb{E}_{\mathbf{z}\sim P_{\mathbf{z}}(\mathbf{z})}[\log(1 - D(G(\mathbf{z}|\mathbf{y})))]. \quad (46)$$

In addition, cGANs can be adapted for regression tasks by encoding the regression labels (i.e., model observations) as additional conditioning information (Figure 3). The cGAN generator takes random noise variables as the original GAN, together with arbitrary model observations as an additional input group. The goal is to generate data samples that correspond to the model observations. The discriminator, which tries to distinguish between real and generated data samples, will now take both real data samples $\mathbf{x}$ and generated data samples $\mathbf{g}$ as input. During training, the cGAN generator learns to map the random noise variables and model observations to corresponding data samples, which helps in regression tasks by generating samples consistent with the given target values. This approach offers a way to tackle problems with complex relationships between inputs and targets, and incorporate the adversarial feature of GANs into regression processes. In this work, $\mathbf{x}$ represents the eVTOL optimal takeoff trajectory control points to be predicted, $\mathbf{y}$ includes design requirements, and we set $\mathbf{z}$ with only one element and assign a Uniform (0, 1) distribution to $\mathbf{z}$. During regression tasks, an average of predictions over a fixed set of $\mathbf{y}$ and 100 randomly sampled $\mathbf{z}$ is used as the predicted model response [39], which in this work, is the optimal takeoff trajectory ($\mathbf{x}$). The network architecture settings for the generator and discriminator within the cGAN follow the same architecture as the regGAN, which will be discussed in Section 2.5.3.

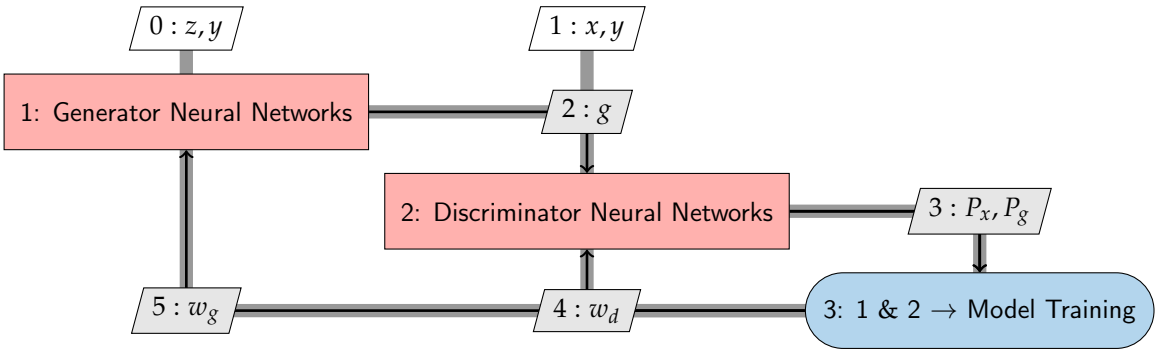

Figure 3. cGAN has similar structures to GAN (shown in Figure 2). cGANs take design requirements ($\mathbf{y}$) and random noise variables ($\mathbf{z}$) as input for the generator network. The time-sequence optimal takeoff trajectories ($\mathbf{x}$) and generated data $\mathbf{g}$ are considered inputs for the discriminator.

2.5.3. Regression Generative Adversarial Networks

To further investigate GAN models for regression tasks, we developed the regGAN for surrogate modeling. The regGAN (Figure 4) shares a similar structure as the cGAN (Figure 3), but the regGAN has only design requirements ($\mathbf{y}$) as the generator input. Moreover, training regGAN incorporates a combined loss function of MSE (Equation (37)) and BC (Equation (38)), such that the regGAN couples a predictive feature of surrogate models and a generative feature of GAN models. The combined loss function is applied in model training.

$$\min_G \max_D \mathcal{L}_{\text{regGAN}}(D,G) = \min(\mathcal{L}_{\text{context}} + \mathcal{L}_{\text{adversarial}}) \tag{47}$$

$$= w_{\text{BC}} \cdot \left(\min_G \max_D \mathcal{L}_{\text{BC}}(D,G) \right) + w_{\text{MSE}} \cdot \left(\min_G \mathcal{L}_{\text{MSE}}(G) \right) \tag{48}$$

$$= w_{\text{BC}} \cdot \min_G \max_D \Big(\mathbb{E}_{\mathbf{x}\sim P_{\text{data}}}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{y}\sim P_{\mathbf{y}}}[\log(1 - D(G(\mathbf{y})))] \Big) \tag{49}$$

$$+ w_{\text{MSE}} \cdot \min_G \frac{1}{N} \sum_{i=1}^{N} (|\mathbf{x}_i - G(\mathbf{y})_i|_2)^2. \tag{50}$$

where w_{BC} and w_{MSE} are constant weights on BC and MSE loss functions; we conduct a parametric study in Section 3 for the best predictive performance.

The MSE loss (also known as context loss) guides the regGAN predictions to match model observations in a similar way as traditional surrogates, while the BC loss (also known as adversarial loss due to the competitive training on GAN models) on the discriminator engenders similar patterns between predicted results and observations. In this work, we leverage the regGAN for eVTOL takeoff trajectory inverse mapping, i.e., directly from design requirements to optimal takeoff trajectories. Thus, the regGAN generator reads the design requirements ($\mathbf{y}$) and predicts the corresponding optimal takeoff trajectories ($\mathbf{x}$). Meanwhile, the discriminator attempts to distinguish between predicted/generated trajectories and actual optimal takeoff trajectories. Thus, the nature of the eVTOL optimal takeoff trajectory prediction makes it necessary to develop and introduce the regGAN surrogate. First, the regGAN makes use of the predictive capability of the DNN via the generator. Second, the optimal takeoff trajectory profiles typically follow realistic patterns. For instance, optimal power and wing angle profiles follow ascending trends in general, meaning that the battery provides more and more power to balance the weight and gradually moves forward by turning the wing from vertical to horizontal positions. In this way, the BC loss based on the GAN architecture facilitates the training by automatically filtering out unrealistic trajectory profiles.

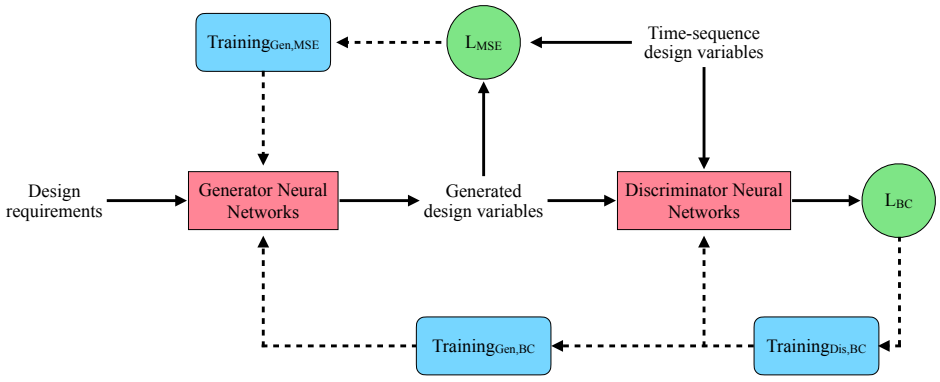

Figure 4. regGAN includes a generator and a discriminator as the original GAN and cGAN. The generator generates time-sequence design variables based on corresponding design requirements through DNN. The discriminator takes generated optimal designs and true time-sequence optimal trajectories in order to distinguish the differences. This results in adversarial competition, which causes an adversarial loss function (using binary crossentropy, $\mathcal{L}_{\text{BC}}$) to train the generator and the discriminator. Additionally, we minimize the mean square error between generated and true design variables ($\mathcal{L}_{\text{MSE}}$) to explicitly train the generator for regression tasks.

The regGAN generator architecture setting is as follows (Table 1). The generator consists of one input layer, two hidden layers, and one output layer. Five neurons in the input layer correspond to five different design requirements. Both hidden layers have 100 neurons, followed by `relu` activation functions. The output layer has twenty-one

neurons corresponding to the twenty-one time-sequence optimal design variables or one neuron for time prediction, followed by `sigmoid` activation functions. The discriminator includes one input layer, two hidden layers, and one output layer. The input layer has 21 neurons corresponding to the time-sequence optimal design variables. Both hidden layers have 100 neurons with `relu` activation functions. The output layer has only one neuron followed by `sigmoid` activation functions in order to distinguish between generated and true optimal design variables. The observations in training data sets by the generator are scaled within the range of [0, 1] via the `MinMaxScaler` within `Scikit-learn`. We finish training the regGAN using the Adam optimizer (Section 2.4.2) within `Tensorflow` with the momentum coefficients of $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The learning rates for the generator and the discriminator networks are both 0.001. The batch size is 20 and the maximum number of training iterations is 1000. We use the same architectures and setups to train three separate regGAN surrogates to predict the power profile, wing angle profile, and total takeoff time. The same architectures and setups also apply to cGAN surrogates for fair comparisons.

Table 1. Neural architectures and training setups of the generator and the discriminator within regGAN.

	Generator	**Discriminator**
Input layer	5 neurons, no activation	21 neurons, no activation
Hidden layer 1	100 neurons, `relu` activation	100 neurons, `relu` activation
Hidden layer 2	100 neurons, `relu` activation	100 neurons, `relu` activation
Output layer	21 neurons/1 neuron, `sigmoid` activation	1 neurons, `sigmoid` activation
Training algorithm	Adam optimizer	Adam optimizer
Training parameters	$\beta_1 = 0.9$, $\beta_2 = 0.999$	$\beta_1 = 0.9$, $\beta_2 = 0.999$
Learning rate	0.001	0.001
Batch size	20	20
Epochs	1000	1000

2.6. Verification Metric

In this work, we implement the mean L_1 relative error ($\bar{\epsilon}_{L_1}$) to evaluate the surrogate predictive performance. The L_1 relative error, also known as the mean absolute percentage error, is a metric used to measure the relative L_1 norm difference between predictions compared with actual values.

$$\bar{\epsilon}_{L_1} = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \frac{\left| \mathbf{y}_{\text{pred},i} - \mathbf{y}_{\text{true},i} \right|}{\left| \mathbf{y}_{\text{true},i} \right|} \times 100\%, \tag{51}$$

where N_{test} is the number of testing samples, $y_{\text{true},i}$ and $y_{\text{pred},i}$ are the true observations and predicted values of the i^{th} set of design requirements. The relative accuracy is calculated as

$$ACC_{L_1} = 1 - \bar{\epsilon}_{L_1}. \tag{52}$$

The L_1 relative accuracy measures the average relative match between predicted values and true observations, expressed in percentage. It gives an indication of to what extend surrogate predictions agree with true observations relative to true observations. The percentage exhibits deeper insights through the errors relative to model response magnitudes.

3. Results and Discussion

In this section, we formulate the optimization problem and vary the operating parameter bounds to consider various takeoff scenarios. We showcase and compare the optimal takeoff trajectory predictions by MOGP, cGAN, and regGAN surrogates with

simulation-based optimal counterparts. The MOGP represents the traditional surrogates (such as polynomial chaos expansions) and shows promising capability in handling multiple outputs. The cGAN makes use of the GAN architecture for regression tasks that share similar principles as the regGAN. Note that the training and testing data sets are generated by simulation-based optimal designs, and the testing accuracy was calculated based on the testing data set. In addition, we do not consider buildings, although the application scenario is set up for UAM, since we mainly focus on the UAM scope of precise transportation/delivery through optimal energy-efficient takeoff designs.

Table 2 formulates the trajectory optimization problem. The objective is to minimize the electrical power consumed to reach a minimum vertical displacement of 305 m and a minimum horizontal speed of 67 m/s. The design requirements (i.e., design constraints and flight condition parameters) include an angle of attack constraint $\alpha_{\text{lim}} \in [10, 15]$ deg, maximum acceleration magnitude $a_{\text{max}} \in [0.2, 0.4]g$ (g is gravitational acceleration), propeller-induced velocity factor $k_{\text{in}} \in [0.3, 1.0]$, electrical and mechanical loss factor $k_{\text{elec}} \in [0.7, 0.9]$, and wing size factor $S_{\text{ref}} \in [0.9, 1.0]$. Note that the a_{max} constraint is a concept to consider for passenger comfort for future real-world transportation, although eVTOL aircraft currently have not been widely applied for such tasks yet. The design variables are the time-sequence electrical power ($\mathbf{P}$) and wing angle to vertical ($\boldsymbol{\theta}$), both of which have 21 cubic curve control points and a total takeoff time (t_{flight}). As mentioned in Section 2, we use the open-source `Dymos` package within `OpenMDAO`. Figure 5 shows the optimal takeoff trajectory profiles, which verify the `Dymos` package in terms of the time history of design variables and takeoff conditions.

Table 2. Trajectory optimization problem formulation.

	Function or Variable	**Description**	**Quantity**
minimize	E	Electrical energy consumed	
w.r.t.	$\mathbf{P}$	Electric power using 21 cubic curve control points	21
	$\boldsymbol{\theta}$	Wing angle to vertical using 21 cubic curve control points	21
	t_{flight}	Takeoff time	1
		Total design variables	**43**
subject to	$y_{\text{final}} \geq 305$ m	Final vertical displacement constraint	1
	$x_{\text{final}} \leq 1400$ m	Final horizontal displacement constraint	1
	$v_x = 67$ m/s	Final horizontal speed constraint	1
	$y \geq 0$ m	Vertical displacement constraint	1
	$a \leq a_{\text{max}}$	Acceleration constraint	1
	$\alpha \leq \alpha_{\text{lim}}$	Positive stall-angle constraint	1
	$\alpha \geq -\alpha_{\text{lim}}$	Negative stall-angle constraint	1
		Total constraints	**7**
Conditions	k_{in}	Propeller-induced velocity factor	
	α_{lim}	Angle of attack constraint value	
	a_{max}	Maximum acceleration magnitude	
	k_{elec}	Electrical and mechanical loses factor	
	S_{ref}	Wing size factor	

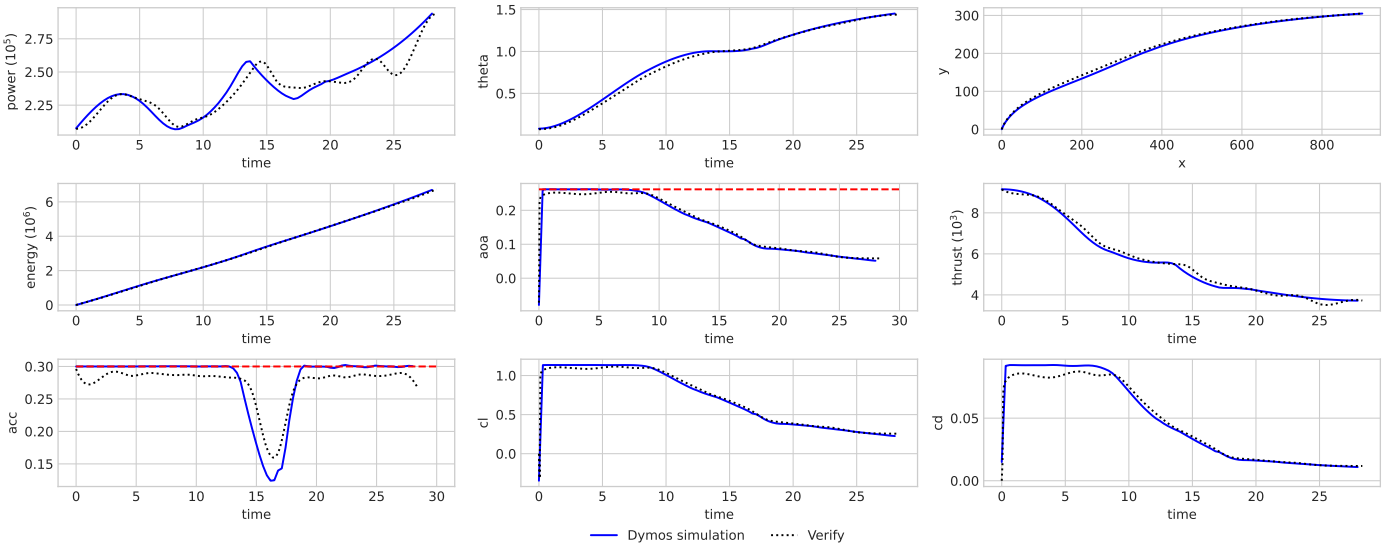

Figure 5. We verified results with `Dymos` package. The blue line represents the direct result of the outputs from the `Dymos` optimization data by NASA, and the black dot line presents the results by Chauhan and Martins [14] as verification data. The units for each y axis are given by the following: power (W), theta (rad), y (m), energy (J), aoa (rad), thrust (N), acc (g), cl (-), cd (-).

3.1. MOGP Surrogate Modeling

We use 1000 random Latin hypercube sampling (LHS) points as training data for MOGP surrogate modeling and 300 LHS testing samples to verify predictive performance. Table 3 shows the mean testing accuracy for each design variable group using different kernel and mean functions. The results show that the SE kernel function has better overall predictive performance than the Matérn kernel. The SE-based MOGP predicts the t_{flight}, $\mathbf{P}$, and $\boldsymbol{\theta}$ at mean testing accuracies of 99.5%, 92.5%, and 94.7%, respectively. The prediction difference between the constant and linear mean functions is negligible.

Table 3. MOGP model conditions and mean ± standard deviation of testing accuracy on testing data set. SE stands for SE kernel function.

Model	Kernel Function	Mean Function	$\bar{\epsilon}_{L_1}$, t_{flight} (%)	$\bar{\epsilon}_{L_1}$, $\mathbf{P}$ (%)	$\bar{\epsilon}_{L_1}$, $\boldsymbol{\theta}$ (%)
M1	SE	Constant	99.6 ± 0.313	92.5 ± 2.93	94.7 ± 2.30
M2	SE	Linear	99.5 ± 0.372	92.5 ± 2.89	94.7 ± 2.32
M3	Matérn	Constant	96.7 ± 2.65	88.7 ± 4.85	94.6 ± 2.20
M4	Matérn	Linear	96.7 ± 2.61	91.4 ± 2.73	94.6 ± 2.20

We chose an arbitrary case to further reveal the predictive performance of each MOGP surrogate. Table 4 shows the design requirements for the visualization case. Table 5 shows the testing accuracy on design variables for MOGP models 1–4. We compare the predicted optimal trajectory profiles by MOGP models 1–4 in Figure 6, where all models achieve similar predictive accuracies. The testing accuracies, shown in Table 5, are all over 90%; however, visualization exhibits obvious discrepancies between surrogate-based and simulation-based optimal trajectories. The results indicate that MOGP surrogates intend to use higher power early but not the maximum power. The early power usage results in higher thrust and greater acceleration in the early takeoff phase by MOGP surrogates than by the simulation-based optimal design. The acceleration by the surrogate-based optimal design in the early phase even violates the maximum acceleration constraint. So, we conclude that the MOGP cannot realize a sufficient accuracy level using 1000 training samples.

Table 4. Design requirements for visualized case.

Design Requirements	Values
Propeller-induced velocity factor, k_{in}	87.75 (%)
Angle of attack constraint, α_{lim}	$\pm$12.25833333 (deg)
Acceleration, a_{max}	0.38766667 (g)
Electrical and mechanical loses factor, k_{elec}	0.761
Wing size, S_{ref}	0.9285

Table 5. Testing accuracy of visualized case for MOGP models 1–4.

Model	$\bar{\epsilon}_{L_1}$, t_{flight} (%)	$\bar{\epsilon}_{L_1}$, P (%)	$\bar{\epsilon}_{L_1}$, θ (%)
M1	99.9	94.4	91.0
M2	99.8	94.7	91.0
M3	95.9	91.5	90.7
M4	95.7	94.3	90.5

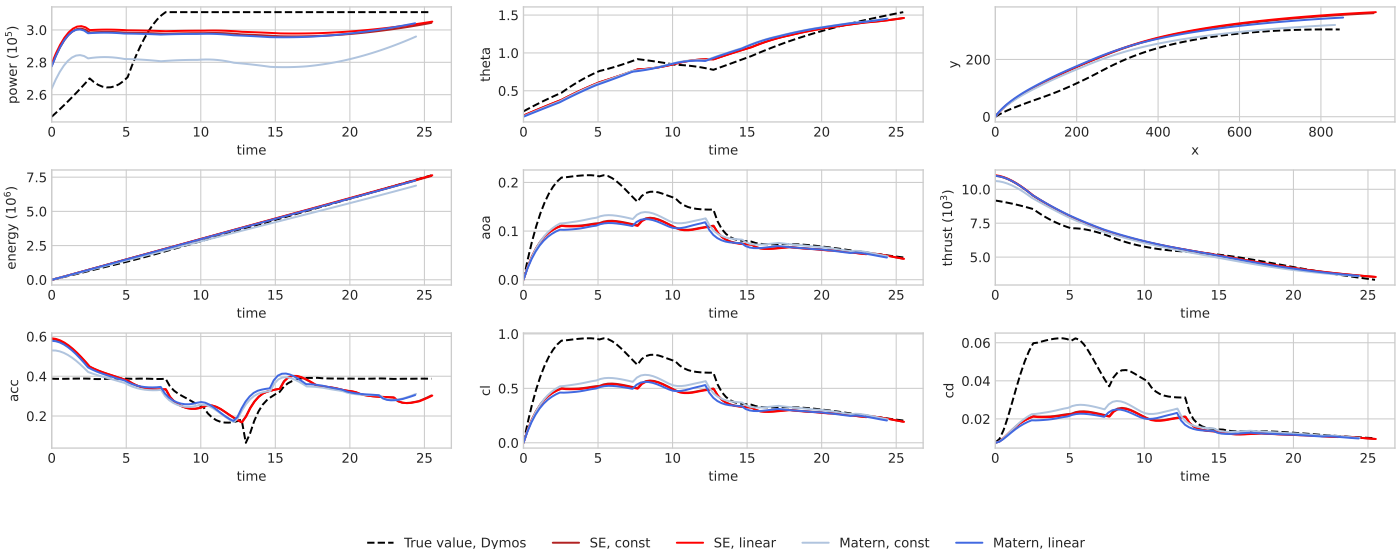

Figure 6. Optimized takeoff trajectory profile for MOGP models 1–4 verified against simulation-based ground truth.

3.2. cGAN Surrogate Modeling

We implement one single-noise cGAN model and compare the predictive performance of this model against the `Dymos` simulation-based optimal trajectory as well as the MOGP predictions shown in Section 3.1. The model architecture and hyperparameter settings are as mentioned in Section 2.5.3. Here, we train the model with MSE and BC loss functions separately. To enable the regression feature with the cGAN, we take the mean value of 100 random noise inputs for each set of design requirements. Table 6 displays the mean and standard deviation of the testing accuracy for cGAN models. The mean of both cGAN models shows that the cGAN has an overall 98% accuracy. The low standard deviation of the cGAN model represents the robustness of the cGAN model. Both BC-based and MSE-based cGAN surrogates outperform the MOGP in terms of accuracy.

We compare and visualize an arbitrary set of predicted results among MSE-based cGAN, BC-based cGAN, and MOGP surrogates using the same design requirements as Table 4. Table 7 presents the testing accuracy of the cGAN models for the visualization case. Figure 7 shows the optimal takeoff trajectory profiles for the cGAN models, with BC and MSE having similar predictive accuracy. The results indicate that the cGAN outperforms the MOGP's predictive accuracy in this visualization case. Specifically, the cGAN captures the general trend of ground truth well due to the predictive power of the DNN surrogate

as well as the addition of the discriminator to drive prediction matching training data set shapes. However, there are still some unexpected wiggles when we look at the acceleration profile. In addition, the cGAN approximates ground truth based on an average of a number of predictions (100 predictions in this work) over the same set of input parameters (design requirements in this work), where the prediction may vary slightly due to Monte Carlo properties. Hence, we develop and introduce the regGAN surrogate for further predictive improvement as follows.

Table 6. cGAN models exhibit better predictive performance over MOGP in terms of mean ± standard deviation in testing accuracy of design variables.

Model	Loss Function	$\bar{\epsilon}_{L_1}$, t_{flight} (%)	$\bar{\epsilon}_{L_1}$, P (%)	$\bar{\epsilon}_{L_1}$, θ (%)
cGAN_{BC}	BC	98.4 ± 0.858	99.0 ± 0.653	98.6 ± 0.424
cGAN_{MSE}	MSE	98.3 ± 0.843	98.8 ± 0.745	98.7 ± 0.426

Table 7. Testing accuracy of visualized case for cGAN model BC and MSE.

Model	Loss Function	$\bar{\epsilon}_{L_1}$, t_{flight} (%)	$\bar{\epsilon}_{L_1}$, P (%)	$\bar{\epsilon}_{L_1}$, θ (%)
cGAN_{BC}	BC	98.8	99.8	98.7
cGAN_{MSE}	MSE	98.9	99.7	98.7

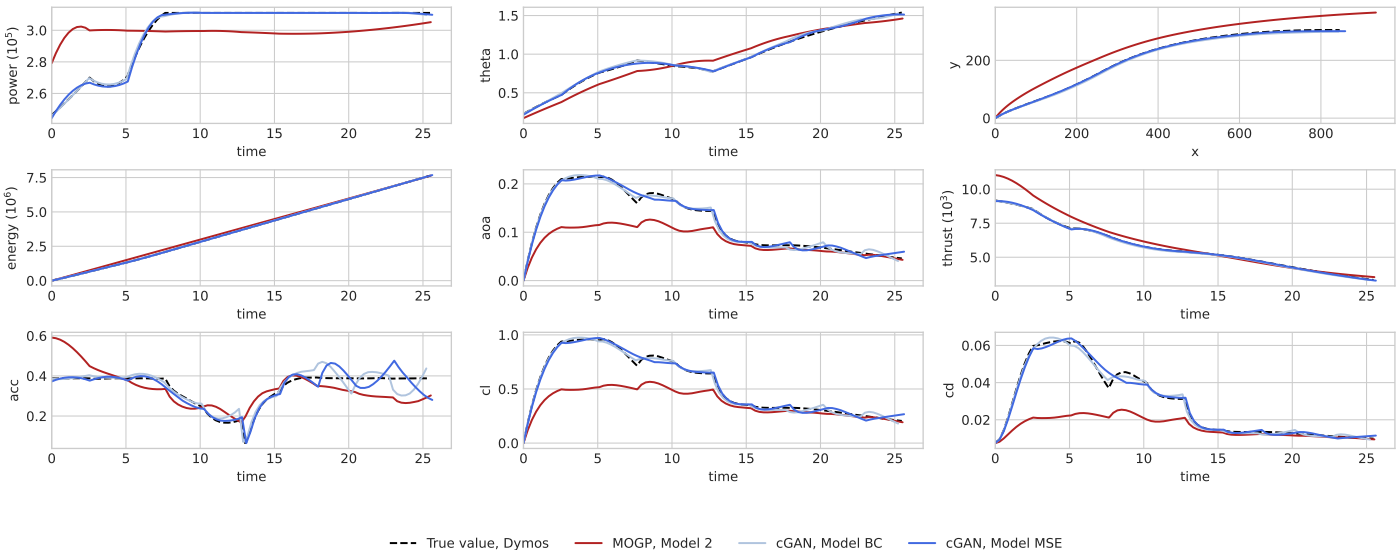

Figure 7. Optimal takeoff trajectory profile comparison on cGAN model BC and MSE and MOGP model 2 with simulation-based ground truth.

3.3. regGAN Surrogate Modeling

The predictive performance of the regGAN surrogates is compared against `Dymos` simulation-based optimal trajectory predictions as well as MOGP and cGAN surrogates. The same 1000 training samples and 300 testing samples are used for regGAN model training and verification, respectively. The architecture and the hyperparameters are introduced in Section 2.5.3. We first focus on the regGAN surrogate with a single MSE loss function by setting a zero weight on BC in the combined loss (CL1) (Table 8). The table shows that the regGAN CL1 model reaches 99.5% accuracy with robust predicted results, which can be recognized by the low standard deviation of testing accuracy. To further compare results, we implement regGAN on the same visualization case in Table 4. Table 9 shows that the regGAN has a mean testing accuracy of over 99.5% in the visualization case. Figure 8 shows that the optimal takeoff trajectory profiles predicted by regGAN match `Dymos` results closer than the cGAN model BC and MSE and MOGP model 2, as expected.

Table 8. Mean ± standard deviation of testing accuracy for regGAN model CL1.

Model	$w_{\mathbf{MSE}}$	$w_{\mathbf{BC}}$	$\bar{\epsilon}_{L_1}$, $t_{\mathbf{flight}}$ (%)	$\bar{\epsilon}_{L_1}$, P (%)	$\bar{\epsilon}_{L_1}$, θ (%)
CL1	1	0	99.5 ± 0.246	99.7 ± 0.196	99.7 ± 0.117

Table 9. Testing accuracy of visualization case for regGAN model BC and MSE.

Model	$w_{\mathbf{MSE}}$	$w_{\mathbf{BC}}$	$\bar{\epsilon}_{L_1}$, $t_{\mathbf{flight}}$ (%)	$\bar{\epsilon}_{L_1}$, P (%)	$\bar{\epsilon}_{L_1}$, θ (%)
CL1	1	0	99.4	99.8	99.8

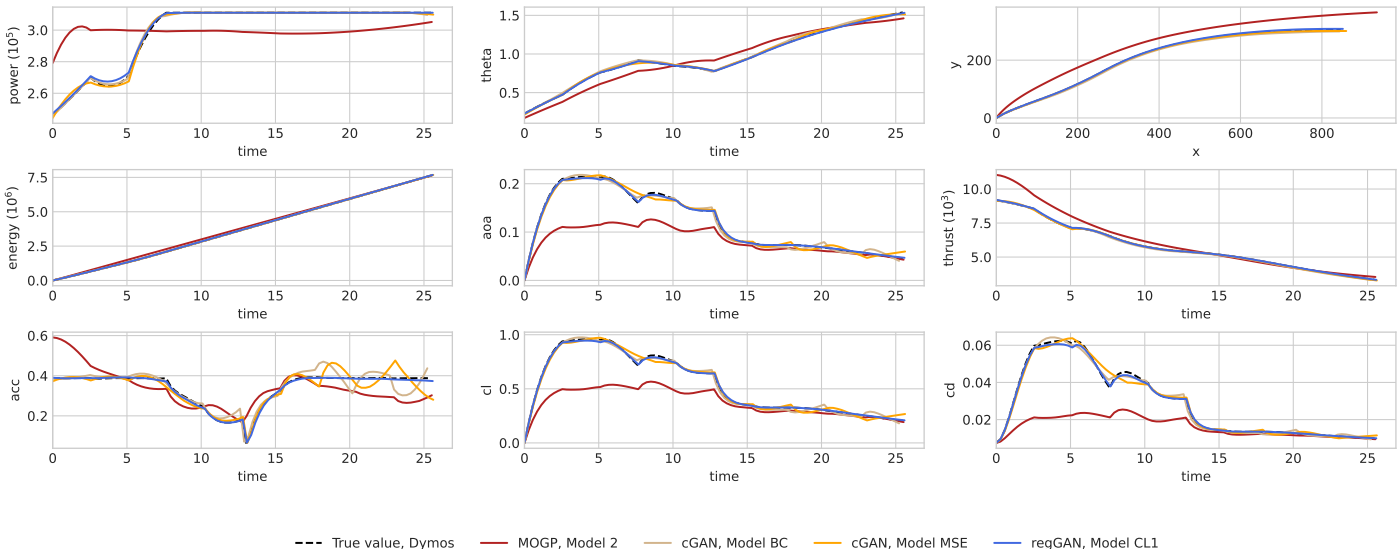

Figure 8. Optimal takeoff trajectory profile for regGAN model CL1 matches the ground truth and outperforms cGAN models BC and MSE and MOGP model 2.

To investigate the regGAN surrogate's performance, we utilized a combination of two loss functions during the model training. Table 10 shows the loss weights together with the mean and standard deviation of testing accuracy for each regGAN model. The results indicate that all regGAN CL models have over 99.6% mean testing accuracy and high predictive robustness, as revealed by the low standard deviations. Table 11 shows that the testing accuracy by regGAN models CL2–4 on the visualization case agrees well with the corresponding mean testing accuracy. Figure 9 shows that all predicted trajectories by regGAN models CL2–4 match the reference profiles, meaning that an accuracy of 99.6% is reliable in lieu of simulation models. regGAN models CL2–4 slightly outperform regGAN model CL1, but the matches towards reference profiles are at a comparable level.

Table 10. regGAN model conditions and mean ± standard deviation of testing accuracy for design variables.

Model	$w_{\mathbf{MSE}}$	$w_{\mathbf{BC}}$	$\bar{\epsilon}_{L_1}$, $t_{\mathbf{flight}}$ (%)	$\bar{\epsilon}_{L_1}$, P (%)	$\bar{\epsilon}_{L_1}$, θ (%)
CL2	1	0.01	99.7 ± 2.388	99.7 ± 0.118	99.7 ± 0.123
CL3	1	0.001	99.7 ± 0.193	99.7 ± 0.160	99.7 ± 0.124
CL4	1	0.0001	99.6 ± 0.216	99.7 ± 0.145	99.7 ± 0.128

Table 11. Testing accuracy for visualization case of regGAN model CL2–4.

Model	w_{MSE}	w_{BC}	$\bar{\epsilon}_{L_1}$, t_{flight} (%)	$\bar{\epsilon}_{L_1}$, P (%)	$\bar{\epsilon}_{L_1}$, θ (%)
CL2	1	0.01	99.6	99.9	99.8
CL3	1	0.001	99.9	99.8	99.8
CL4	1	0.0001	99.9	99.9	99.5

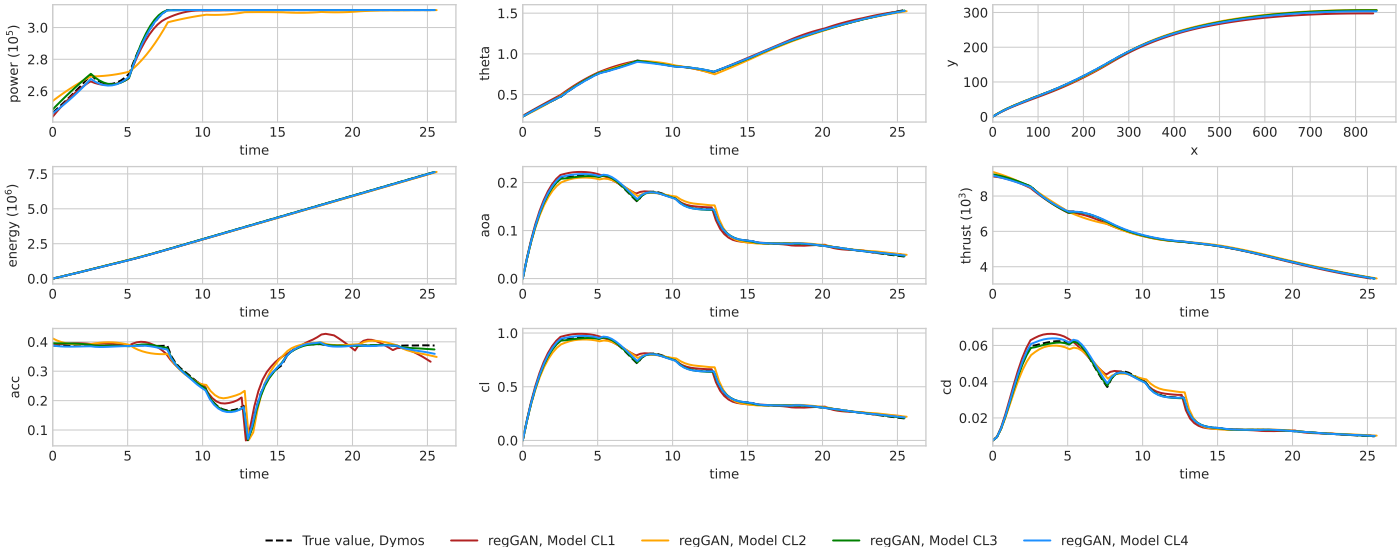

Figure 9. Optimal takeoff trajectory profiles predicted by regGAN models CL1–4 match well with simulation-based ground truth.

We also conducted a parametric study for different combined loss weights to further investigate regGAN performance. We use a different number of training samples, ranging from 50, 100, 200, 400, 600, 800, to 1000. Table 12 indicates that using 100 and 200 training samples is able to obtain the above with an overall 97% accuracy. Note that when provided with 50 training samples, CL1 has the lowest mean testing accuracy and highest standard deviation time t_{flight} prediction, while CL2, CL3, and CL4 have better overall predictive performance. Moreover, regGAN model CL2 ($w_{\text{MSE}} = 1$ and $w_{\text{BC}} = 0.01$) achieves over 99.5% accuracy starting with 400 training samples, while all the other regGAN surrogates require at least 800 training samples. In addition, regGAN model CL2 consistently shows lower predictive standard deviations (such as 0.251%, 0.257%, and 0.145% on t_{flight}, **P**, and $\boldsymbol{\theta}$) on the testing data set, further confirming a better and more robust predictive performance. Based on the visualization case, a mean testing accuracy of over 99.5% can be considered sufficiently accurate with negligible differences.

In sum, the MOGP surrogates could not match ground truths or capture the general trend of optimal takeoff trajectories well using 1000 training samples, which may be due to the Gaussian assumption of GP series models. The cGAN surrogates achieve better performance over MOGP using the 1000 training samples, with a closer match towards the general trend of ground truth labels, mainly because of the guidance of the discriminator for generating similar data patterns as the training data set. The regGAN surrogates outperform MOGP and cGAN since regGAN is trained with a combined loss function of MSE and BC adversarial losses. The BC adversarial loss leads regGAN to handle the general trend of observations, while the MSE loss directly drives the match between predictions and observations.

Table 12. Parametric study for regGAN models exhibits deeper insights on mean ± standard deviation of testing accuracy with respect to the number of training samples for each design variable group. Note that we only vary wBC while keeping w_{MSE} as 1.

Model	$w_{\mathbf{BC}}$	Samples	$\bar{\epsilon}_{L_1}$, t_{flight} (%)	$\bar{\epsilon}_{L_1}$, P (%)	$\bar{\epsilon}_{L_1}$, θ (%)
CL1	0	50	95.2 ± 0.279	97.4 ± 1.58	98.3 ± 0.758
		100	98.6 ± 0.906	98.5 ± 1.05	98.9 ± 0.414
		200	99.3 ± 0.368	99.2 ± 0.461	99.4 ± 0.212
		400	99.5 ± 0.314	99.5 ± 0.389	99.6 ± 0.167
		600	99.2 ± 0.477	99.4 ± 0.294	99.6 ± 0.148
		800	99.7 ± 0.281	99.7 ± 0.146	99.7 ± 0.119
		1000	99.5 ± 0.246	99.7 ± 0.196	99.7 ± 0.117
CL2	0.01	50	97.4 ± 1.51	97.7 ± 0.852	97.6 ± 0.955
		100	97.4 ± 0.732	98.6 ± 0.829	98.9 ± 0.434
		200	99.3 ± 0.443	97.8 ± 0.980	98.9 ± 0.427
		400	99.6 ± 0.251	99.6 ± 0.257	99.6 ± 0.145
		600	99.6 ± 0.204	99.7 ± 0.215	99.6 ± 0.153
		800	99.5 ± 0.314	99.7 ± 0.158	99.6 ± 0.147
		1000	99.7 ± 0.239	99.7 ± 0.118	99.7 ± 0.123
CL3	0.001	50	95.3 ± 2.77	97.7 ± 1.29	98.0 ± 0.759
		100	97.2 ± 1.72	96.4 ± 2.08	96.6 ± 1.31
		200	99.3 ± 0.380	99.0 ± 0.557	99.1 ± 0.308
		400	99.6 ± 0.343	99.4 ± 0.378	99.4 ± 0.230
		600	99.7 ± 0.228	98.9 ± 0.767	99.4 ± 0.239
		800	99.5 ± 0.228	99.7 ± 0.194	99.7 ± 0.127
		1000	99.7 ± 0.193	99.7 ± 0.160	99.7 ± 0.124
CL4	0.0001	50	97.1 ± 1.64	98.5 ± 1.11	96.4 ± 1.78
		100	98.5 ± 0.757	98.9 ± 0.648	97.7 ± 0.847
		200	99.2 ± 0.384	98.8 ± 0.706	98.9 ± 0.494
		400	99.4 ± 0.346	99.4 ± 0.358	99.3 ± 0.249
		600	99.2 ± 0.611	99.3 ± 0.509	99.4 ± 0.195
		800	99.6 ± 0.287	99.5 ± 0.363	99.5 ± 0.182
		1000	99.6 ± 0.216	99.7 ± 0.145	99.7 ± 0.128

4. Conclusions

In this paper, we investigated surrogate-based optimal takeoff trajectory predictions for electric vertical takeoff and landing (eVTOL) drones within the scope of urban air mobility. We developed the regression generative adversarial network (regGAN), which outperformed the Gaussian process (MOGP) and the conditional generative adversarial network (cGAN) by achieving over 99.5% accuracy. We summarize the main contribution of this work as follows.

First, we implemented the surrogate-based inverse mapping concept into eVTOL optimal trajectory design for the first time. Specifically, surrogate models took design requirements as input and predicted optimal trajectories. We realized fast interactive eVTOL takeoff trajectory design without running any optimizations since the trained surrogates directly predicted optimal trajectories. However, reducing training costs is essential since each training sample requires a simulation-based trajectory design optimization.

Second, we introduced the MOGP, a representative traditional surrogate model, into the eVTOL takeoff trajectory design for rapid predictions. The results showed that the MOGP with a square exponential kernel function could accurately capture the inverse mapping using 1000 training samples. We then implemented a cGAN for eVTOL inverse mapping since cGAN also makes use of the GAN architecture for regression tasks. The results revealed that the cGAN achieved over 98% generalization accuracy in predicting optimal designs using the same 1000 training samples as the MOGP, which means the cGAN outperformed the MOGP on predictive performance (around 92% accuracy). In addition, the visualization case verified that the cGAN could match the general trend of optimal designs well with actual observations from `Dymos` but missed detailed features.

Third, we introduced the regGAN into takeoff trajectory design for the first time and achieved over 99.6% accuracy in predicting optimal design variables with the same 1000 training samples as the MOGP and cGAN. By varying the weights of different loss functions, the regGAN could achieve over 99.6% accuracy. Moreover, results indicated that the best regGAN surrogate architecture consistently achieved over 99.5% accuracy if provided with 400 or more random training samples. This confirmed the outstanding predictive performance and potential generality of the regGAN.

In future work, we are planning to explore and develop other novel deep learning architectures in regGAN. In addition, the simulation models used in this work are not high-fidelity models but effective for describing the physics; we will increase the fidelity of simulation models in future work, which may lead to a higher training cost for surrogate modeling. Moreover, we will consider takeoff time as another constraint to make sure the total takeoff will not take unreasonable time.

Author Contributions: Conceptualization, X.D.; methodology, X.D.; software, S.-T.Y.; validation, S.-T.Y.; investigation, S.-T.Y.; resources, X.D.; data curation, S.-T.Y.; writing—original draft preparation, S.-T.Y.; writing—review and editing, X.D.; visualization, S.-T.Y.; supervision, X.D.; project administration, X.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are available on request from the corresponding author. The data are not publicly available because the authors currently have continuous research to conduct on these data.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Wikipedia. EHang. 2023. Available online: https://en.wikipedia.org/wiki/EHang (accessed on 27 June 2023).
2. Wisk Aero LLC. Advanced Air Mobility Presents Opportunity to Bring Economic, Social, and Environmental Benefits to South East Queensland. *Tech. Rep.* 2023. Available online: https://wisk.aero/news/press-release/advanced-air-mobility-presents-opportunity-to-bring-economic-social-and-environmental-benefits-to-south-east-queensland/ (accessed on 15 January 2022).
3. Johnson, W.; Silva, C.; Solis, E. Concept Vehicles for VTOL Air Taxi Operations. In Proceedings of the AHS Technical Conference on Aeromechanics Design for Transformative Vertical Flight, San Francisco, CA, USA, 16–18 January 2018.
4. Bacchini, A.; Cestino, E. Electric VTOL Configurations Comparison. *Aerospace* **2019**, *6*, 26. [CrossRef]
5. Na, Z.; Liu, Y.; Shi, J.; Liu, C.; Gao, Z. UAV-Supported Clustered NOMA for 6G-Enabled Internet of Things: Trajectory Planning and Resource Allocation. *IEEE Internet Things J.* **2021**, *8*, 15041–15048. [CrossRef]
6. Hua, B.; Ni, H.; Zhu, Q.; Wang, C.-X.; Zhou, T.; Mao, K.; Bao, J.; Zhang, X. Channel Modeling for UAV-to-Ground Communications With Posture Variation and Fuselage Scattering Effect. *IEEE Trans. Commun.* **2023**, *71*, 3103–3116. [CrossRef]
7. Boelens, J.-H. Pioneering the Urban Air Taxi Revolution. *Tech. Rep.* 2019. Available online: https://www.volocopter.com/urban-air-mobility/ (accessed on 15 January 2022).
8. Electric VTOL News. Joby Aviation S4 2.0 (prototype). 2023 Available online: https://evtol.news/joby-s4 (accessed on 27 June 2023).
9. Electric VTOL News. Aurora Flight Sciences Pegasus PAV. 2019. Available online: https://evtol.news/aurora/ (accessed on 27 June 2023).
10. Wikipedia. Airbus A^3 Vahana. 2022. Available online: https://en.wikipedia.org/wiki/Airbus_A%C2%B3_Vahana (accessed on 27 June 2023).
11. Yeh, S.-T.; Yan, G.; Du, X. Inverse Machine Learning Prediction for Optimal Tilt-Wing eVTOL Takeoff Trajectory. *Aiaa Aviat. 2023 Forum* **2023**, *2023*, 3593. [CrossRef]
12. Pradeep, P.; Wei, P. Energy Optimal Speed Profile for Arrival of Tandem Tilt-Wing eVTOL Aircraft with RTA Constraint. In Proceedings of the IEEE/CSAA Guidance, Navigation and Control Conference (GNCC), Xiamen, China, 10–12 August 2018.
13. Pradeep, P.; Wei, P. Energy Efficient Arrival with RTA Constraint for Urban eVTOL Operations. In Proceedings of the 2018 AIAA Aerospace Sciences Meeting, Kissimmee, FL, USA, 8–12 January 2018.
14. Chauhan, S.S.; Martins, J.R.R.A. Tilt-wing eVTOL takeoff trajectory optimization. *J. Aircr.* **2020**, *57*, 93–112. [CrossRef]
15. Li, J.; Du, X.; Martins, J.R. Machine learning in aerodynamic shape optimization. *Prog. Aerosp. Sci.* **2022**, *134*, 100849. [CrossRef]
16. Koziel, S.; Leifsson, L. Surrogate-Based Aerodynamic Shape Optimization by Variable-Resolution Models. *AIAA J.* **2013**, *51*, 94–106. [CrossRef]

17. Nagawkar, J.; Leifsson, L. Applications of Polynomial Chaos-Based Cokriging to Simulation-Based Analysis and Design Under Uncertainty. In Proceedings of the International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Online, 17–19 August 2020.
18. Lobo do Vale, J.; Sohst, M.; Crawford, C.; Suleman, A.; Potter, G.; Banerjee, S. On the multi-fidelity approach in surrogate-based multidisciplinary design optimisation of high-aspect-ratio wing aircraft. *Aeronaut. J.* **2023**, *127*, 2–23. [CrossRef]
19. Iuliano, E.; Quagliarella, D. Aerodynamic shape optimization via non-intrusive POD-based surrogate modelling. In Proceedings of the 2013 IEEE Congress on Evolutionary Computation, Cancun, Mexico, 20–23 June 2013; pp. 1467–1474. [CrossRef]
20. Li, S.; Trevelyan, J.; Wu, Z.; Lian, H.; Wang, D.; Zhang, W. An adaptive SVD–Krylov reduced order model for surrogate based structural shape optimization through isogeometric boundary element method. *Comput. Methods Appl. Mech. Eng.* **2019**, *349*, 312–338. [CrossRef]
21. Singh, P.; Couckuyt, I.; Ferranti, F.; Dhaene, T. A constrained multi-objective surrogate-based optimization algorithm. In Proceedings of the 2014 IEEE Congress on Evolutionary Computation (CEC), Beijing, China, 6–11 July 2014; pp. 3080–3087. [CrossRef]
22. Shen, Y.; Huang, W.; Yan, L.; Zhang, T.-T. Constraint-based parameterization using FFD and multi-objective design optimization of a hypersonic vehicle. *Aerosp. Sci. Technol.* **2020**, *100*, 105788. [CrossRef]
23. Li, M.; Wang, Z. Surrogate model uncertainty quantification for reliability-based design optimization. *Reliab. Eng. System Saf.* **2019**, *192*, 106432. [CrossRef]
24. Du, X.; Leifsson, L.; Koziel, S.; Bekasiewicz, A. Airfoil Design Under Uncertainty Using Non-Intrusive Polynomial Chaos Theory and Utility Functions. *Procedia Comput. Sci.* **2017**, *108*, 1493–1499. [CrossRef]
25. Shao, J.; Shi, L.; Cheng, Y.; Li, T. Asynchronous Tracking Control of Leader–Follower Multiagent Systems With Input Uncertainties Over Switching Signed Digraphs. *IEEE Trans. Cybern.* **2022**, *52*, 6379–6390. [CrossRef] [PubMed]
26. Li, W.; Qin, K.; Li, G.; Shi, M.; Zhang, X. Robust bipartite tracking consensus of multi-agent systems via neural network combined with extended high-gain observer. *ISA Trans.* **2023**, *136*, 31–45. [CrossRef] [PubMed]
27. Rasmussen, C.E.; Williams, C.K.I. *Gaussian Processes for Machine Learning*; Adaptive Computation and Machine Learning; MIT Press: Cambridge, MA, USA, 2006.
28. Liu, M.; Chowdhary, G.; Castra da Silva, B.; Liu, S.-Y.; How, J.P. Gaussian Processes for Learning and Control: A Tutorial with Examples. *IEEE Control Syst. Mag.* **2018**, *38*, 53–86. [CrossRef]
29. Ver Hoef, J.M.; Barry, R.P. Constructing and fitting models for cokriging and multivariable spatial prediction. *J. Stat. Plan. Inference* **1998**, *69*, 275–294. [CrossRef]
30. Chiles, J.-P.; Delfiner, P. *Geostatistics: Modeling Spatial Uncertainty*; John Wiley & Sons: Hoboken, NJ, USA, 2012; Volume 713.
31. Thelen, A.S.; Bryson, D.E.; Stanford, B.K.; Beran, P.S. Multi-Fidelity Gradient-Based Optimization for High-Dimensional Aeroelastic Configurations. *Algorithms* **2022**, *15*, 131. [CrossRef]
32. Tao, J.; Sun, G. Application of Deep Learning Based Multi-Fidelity Surrogate Model to Robust Aerodynamic Design Optimization. *Aerosp. Sci. Technol.* **2019**, *92*, 722–737. [CrossRef]
33. Renganathan, S.A.; Maulik, R.; Ahuja, J. Enhanced data efficiency using deep neural networks and Gaussian processes for aerodynamic design optimization. *Aerosp. Sci. Technol.* **2021**, *111*, 106522. [CrossRef]
34. O'Leary-Roseberry, T.; Du, X.; Chaudhuri, A.; Martins, J. R.; Willcox, K.; Ghattas, O. Learning high-dimensional parametric maps via reduced basis adaptive residual networks. *Comput. Methods Appl. Mech. Eng.* **2022**, *402*, 115730. [CrossRef]
35. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems 27*; Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q., Eds.; MIT Press: Cambridge, MA, USA, 2014; pp. 2672–2680.
36. Chen, W.; Chiu, K.; Fuge, M.D. Airfoil design parameterization and optimization using bézier generative adversarial networks. *AIAA J.* **2020**, *58*, 4723–4735. [CrossRef]
37. Du, X.; He, P.; Martins, J.R.R.A. A B-Spline-based Generative Adversarial Network Model for Fast Interactive Airfoil Aerodynamic Optimization. In *AIAA SciTech Forum*; AIAA: Orlando, FL, USA, 2020. [CrossRef]
38. Du, X.; Martins, J.R. Super Resolution Generative Adversarial Networks for Multi-Fidelity Pressure Distribution Prediction. In *AIAA SCITECH 2023 Forum*; AIAA: Orlando, FL, USA, 2023; p. 0533.
39. Mirza, M.; Osindero, S. Conditional Generative Adversarial Nets. *arXiv* **2014**, arXiv:1411.1784. 2014.
40. Aggarwal, K.; Kirchmeyer, M.; Yadav, P.; Keerthi, S.S.; Gallinari, P. Regression with conditional gan. *arXiv* **2019**, arXiv:1905.12868v1.
41. Ye, K.; Wang, Z.; Chen, P.; Piao, Y.; Zhang, K.; Wang, S.; Jiang, X.; Cui, X. A novel GAN-based regression model for predicting frying oil deterioration. *Sci. Rep.* **2022**, *12*, 10424. [CrossRef]
42. Falck, R.; Gray, J.S.; Ponnapalli, K.; Wright, T. dymos: A Python package for optimal control of multidisciplinary systems. *J. Open Source Softw.* **2021**, *6*, 2809. [CrossRef]
43. Gray, J.S.; Hwang, J.T.; Martins, J.R.R.A.; Moore, K.T.; Naylor, B.A. OpenMDAO: An open-source framework for multidisciplinary design, analysis, and optimization. *Struct. Multidiscip.* **2019**, *59*, 1075–1104. [CrossRef]
44. de Wolff, T. Cuevas, A.; Tobar, F. MOGPTK: The Multi-Output Gaussian Process Toolkit. *Neurocomputing* **2020**, *424*, 49–53. [CrossRef]

45. Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G.S.; Davis, A.; Dean, J.; Devin, M.; et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *arXiv* **2016**, arXiv:1603.04467.
46. Paris, S.; Riehl, J.; Sjauw, W. Enhanced procedures for direct trajectory optimization using nonlinear programming and implicit integration. In Proceedings of the AIAA/AAS Astrodynamics Specialist Conference and Exhibit, Keystone, CO, USA, 21–24 August 2006; p. 6309.
47. Hargraves, C.R.; Paris, S.W. Direct trajectory optimization using nonlinear programming and collocation. *J. Guid. Dyn.* **1987**, *10*, 338–342. [CrossRef]
48. Schubert, G.R. Algorithm 211: Hermite Interpolation. *Commun. ACM* **1963**, *6*, 617. [CrossRef]
49. Tangler, J.L.; Ostowari, C. Horizontal axis wind turbine post stall airfoil characteristics synthesization. In Proceedings of the DOE/NASA Wind Turbine Technology Workshop, Cleveland, OH, USA, 8–10 May 1984.
50. Glauert, H. Airplane propellers. *Aerodyn. Theory* **1935**, 169–360. [CrossRef]
51. Ypma, T.J. Historical development of the Newton–Raphson method. *SIAM Rev.* **1995**, *37*, 531–551. [CrossRef]
52. McCormick, B.W. *Aerodynamics of V/STOL Flight*, 1st ed.; Academic Press: Cambridge, MA, USA, 1967.
53. Leishman, J.G. *Principles of Helicopter Aerodynamics*, 1st ed.; The Press Syndicate of the University of Cambridge: New York, NY, USA, 2000.
54. Biswas, B.; Chatterjee, S.; Mukherjee, S.; Pal, S. A discussion on Euler method: A review. *Electron. J. Math. Anal. Appl.* **2013**, *1*, 2090–2792.
55. Schulz, E.; Speekenbrink, M.; Krause, A. A tutorial on Gaussian process regression: Modelling, exploring, and exploiting functions. *J. Math. Psychol.* **2018**, *85*, 1–16. [CrossRef]
56. Parra, G.; Tobar, F. Spectral Mixture Kernels for Multi-Output Gaussian Processes. In *Advances in Neural Information Processing Systems*; Guyon, I., Luxburg, U.V.,Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2017; Volume 30. Available online: https://proceedings.neurips.cc/paper_files/paper/2017/file/333cb763facc6ce398ff83845f224d62-Paper.pdf (accessed on 10 January 2022).
57. de G. Matthews, A.G.; van der Wilk, M.; Nickson, T.; Fujii, K.; Boukouvalas, A.; León-Villagrá, P.; Ghahramani, Z.; Hensman, J. GPflow: A Gaussian Process Library using TensorFlow. *J. Mach. Learn.* **2017**, *18*, 1–6.
58. LeCun, Y.; Bengio, Y.; Hinton, G. Deep Learning. *Nature* **2015**, *521*, 436–444. [CrossRef]
59. Agarap, A.F. Deep learning using rectified linear units (relu). *arXiv* **2018**, arXiv:1803.08375.
60. Han, J.; Moraga, C. The influence of the sigmoid function parameters on the speed of backpropagation learning. In *International Workshop on Artificial Neural Networks*; Springer: Berlin/Heidelberg, Germany, 1995; pp. 195–201.
61. Allen, D.M. Mean square error of prediction as a criterion for selecting variables. *Technometrics* **1971**, *13*, 469–475. [CrossRef]
62. Ho, Y.; Wookey, S. The real-world-weight cross-entropy loss function: Modeling the costs of mislabeling. *IEEE Access* **2019**, *8*, 4806–4813. [CrossRef]
63. Qian, N. On the Momentum Term in Gradient Descent Learning Algorithms. *Neural Netw.* **1999**, *12*, 145–151. [CrossRef] [PubMed]
64. Tieleman, S.; Hinton, G. Lecture 6.5—RMSProp: Neural Networks for Machine Learning. *Coursera Tech. Rep.* **2012**, *6*. Available online: https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf (accessed on 10 June 2022).
65. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.